OHJELMOINNIN OPETTAMINEN LAPSILLE

Oulun yliopisto
Tieto- ja sähkötekniikan tiedekunta
Tietojenkäsittelytiede
Kandidaatin tutkielma
Grundström Stefan
22.5.2016

Tiivistelmä

Tämä kandidaatin tutkielma käsittelee ohjelmoinnin opettamista lapsille. Tutkielmassa perehdytään kysymyksiin miksi ja miten ohjelmointia tulisi opettaa. Lisäksi käsitellään lapsen kehityksen vaikutusta ohjelmoinnin opettamistavan valintaan. Valitsin tämän aiheen mielenkiinnon ja ajankohtaisuuden vuoksi. Ohjelmoinnin opettaminen on ajankohtaista, koska sitä tullaan opettamaan osana matematiikkaa syksyllä 2016 peruskoulun opetussuunnitelmassa. Tällä hetkellä on tärkeää pohtia keinoja, miten ohjelmoinnin opettamisen sisällyttäminen on järkevää ja kannattavaa.

Tutkielma käsittelee vaihtoehtoja tavalliselle tekstipohjaiselle ohjelmoinnille, joiden tarkoituksena on helpottaa lasta oppimaan ohjelmointia. Näitä ovat ToonTalk, Scratch, ScratchJr, elektroniset palikat, Stagecast Creator, Blockly, LaPlaya, Logo, TangibleK ja Alice. Nämä kaikki käsitellään tutkielmassa pääpiirteittäin ja esitellään, minkä ikäisille ne ovat suunnattu.

Ohjelmoinnin opettaminen on haastavaa, minkä vuoksi myös näitä haasteita on käsitelty tutkielmassa ja näihin on pyritty löytämään ratkaisuja ja apukeinoja. Tutkielmassa korostuu ohjelmoinnin opettamisen tärkeys muunkin kuin ohjelmointitaitojen kehittämismielessä. Opettamalla ohjelmointia opetetaan myös ajattelua, jolloin lasten ongelmanratkaisukyky ja luovuus kehittyvät. Ohjelmoinnin opettaminen auttaa myös muiden oppiaineiden opettelua ja sitä voidaankin käyttää osana muiden oppiaineiden opetusta.

Avainsanat
ohjelmoinnin opettaminen, lapset, ohjelmointi, ohjelmointiympäristöt, ohjelmointikielet

Ohjaaja
Prof. Netta Iivari

1. Johdanto

On lähes varmaa, että nykyajan opiskelijat tulevat vuorovaikuttamaan teknologian kanssa läpi heidän työelämänsä, huolimatta siitä, minkä uran he aikovat valita. Ohjelmoinnin opettamisen arvoa nuorille oppilaille on yleisesti kannatettu, koska se mahdollistaa lapsia ymmärtämään, mistä ohjelmoinnissa on kyse, ja opetuksen myötä tulevaa laskennallista ajattelua pidetään yleisesti hyödyllisenä oppilaiden tulevaisuuden kannalta. Lähtökohtainen tavoite ei ole varmistaa ihmisiä ammattiohjelmoijiksi, vaan kasvattaa uusi luova sukupolvi järjestelmällisiä ajattelijoita, jotka käyttävät ohjelmointia ilmaisemaan omia ideoitaan. (Duncan 2014.)

On huolestuttavaa, että suurin osa nuorista teini-ikäisistä lapsista on täysin tietämättömiä tietojenkäsittelystä oppiaineena ja uravalintana. Tällainen tietämättömyys yhdistetään usein negatiivisiin asenteisiin tietojenkäsittelyä kohtaan. Tietojenkäsittelyn ongelmana on siis sen imago. Oppilailla ei ole selvää käsitystä tietojenkäsittelystä ja tietojenkäsittelijöistä, minkä vuoksi heidän on vaikea kuvitella itseään sille alalle. Imago-ongelmaan voisi olla ratkaisuna opettaa lapsille mitä tietojenkäsittely on ja kuinka sitä käytetään käytännössä esimerkiksi elämänlaadun parantamiseen ja elokuvien erikoistehosteisiin. (Grover 2015.)

Didaktisen näkökulman kannalta tietokoneohjelmointi on kyky, joka auttaa oppilaita syventämään heidän ymmärrystään monessa tietämyksen alassa. Samalla, kun oppilas yrittää ohjelmoida eli ”opettaa” tietokonetta kuinka ratkaista ongelma, hänellä on mahdollisuus ilmaista ajatuksiaan, nähdä lopputulokset, selkeyttää ajatuksiaan prosessissa ja saada välitöntä palautetta. Ohjelmointi on prosessi, joka auttaa kehittämään ongelmanratkaisu- ja metakognitiivisia taitoja. Myös useat muut tutkimukset ovat osoittaneet ohjelmoinnin positiivisen vaikutuksen kognitiiviseen kehitykseen. Lisäksi on osoitettu, että oppilaat, joilla oli kokemusta ohjelmoinnista, saivat korkeampia tuloksia erilaisissa kognitiivisia kykyjä mittaavissa testeissä verrattuna niihin oppilaisiin, joilla ei ollut yhtään kokemusta ohjelmoinnista. (Fessakis 2013.)

Tietokoneohjelma koostuu sarjoista käskyjä, joita ohjelmoija on laatinut. Käskyt ohjaavat tietokoneen toimintaa tietyssä tehtävässä. Ohjelmointi tarkoittaa siis tietokoneohjelman toimintaohjeiden ja periaatteiden määrittelemistä. Ohjelman luomiseksi tehdään tekstimuotoinen esitys käyttäen jotain ohjelmointikieltä. Esitystä kutsutaan lähdekoodiksi, joka on ihmiselle ymmärrettävässä muodossa. Valmis lähdekoodi täytyy lopuksi kääntää joko kääntäjällä tai tulkilla konekieliseen muotoon, jota tietokone ymmärtää, jotta saadaan lopputuloksena suorituskelpoinen ohjelma. (Tirronen 2011.)

Tietokoneohjelmoinnin taidot voivat tehostaa algoritmista ajattelua (computational thinking). Algoritminen ajattelu sisältää käsitteitä kuten ongelmanratkaisun, suunnittelumenetelmät ja ymmärryksen ihmisen käyttäytymisestä (Garneli 2015). Lisäksi ajattelumalli käsittää ongelmien purkamisen pienemmiksi osiksi ja itse algoritmien luomisen (Liukas 2015). Tämän näkökulman perusteella algoritmista ajattelua voisi suositella kaikille eikä vain tietojenkäsittelijöille. (Garneli 2015).

Alustavan suunnitelman mukaan ohjelmointia aletaan opettaa 1.-6. -luokkalaisille syksyllä 2016, minkä jälkeen edetään vuosi kerrallaan kohti ylempiä luokkia aina yläasteen 9. luokalle asti. Tarkoituksena olisi lopulta opettaa ohjelmointia 1.-9.-luokkalaisille syksystä 2019 lähtien. Opetussuunnitelman muutoksen tarkoituksena on ohjata ja innostaa lapsia ohjelmoinnin pariin, ja kehittää heidän ajattelun taitojaan. Vaikka kaikista lapsista ei tule ammattiohjelmoijia, ohjelmoinnin opettaminen tarjoaa kaikille lapsille tärkeitä taitoja, kuten ongelmien pilkkomisen pienempiin palasiin, mikä helpottaa suurten ja hankalien ongelmien ratkaisemisessa. Ongelmien pilkkomista käytetään usein ohjelmoinnissa, mutta sen osaaminen on yhtä hyödyllistä ja tärkeää myös arkielämän eri tilanteissa. Lähtökohtaisesti ohjelmointia opetetaan matematiikan ohella, mutta sitä yritetään linkittää mahdollisuuksien mukaan myös muihin oppiaineisiin. (Mykkänen & Liukas 2014.)

Tutkielma käsittelee 4-15-vuotiaita lapsia, jotka käyvät ohjattua opetusta päiväkodissa tai peruskoulussa. Tutkielmassa käytetyt tutkimukset ovat kohdistuneet pääasiallisesti amerikkalaisiin lapsiin. Tutkielmassa pyritään selvittämään mitä ohjelmoinnin työkaluja voisi käyttää opettaessa ohjelmointia lapsille. Lisäksi tutkitaan mistä syistä ohjelmointia tulisi opettaan lapsille. Erityistä huomiota saavat lapsen iän ja kehityksen merkitys.

Tutkielmassa esitellään lapsille suunnattuja ohjelmoinnin kieliä ja ympäristöjä. Lapsille suunnitellut työkalut on tehty helpottamaan ohjelmoinnin oppimista, minkä vuoksi kielet ja ympäristöt muodostavat kokonaisuuden, eli ohjelmointiympäristöjä käytetään niille suunnitelluilla ohjelmointikielillä. Tutkielmassa ohjelmointiympäristöt ja –kielet on esitelty pääosin erillään toisistaan omissa kappaleissaan. Lapsille suunnatuilla ohjelmointiympäristöillä on yleensä oma ohjelmointikielensä, minkä vuoksi sekä ohjelmointikieltä että –ympäristöä voidaan käsitellä eri kappaleissa joko ympäristönä tai kielenä tarkoittaen samaa kokonaisuutta.

Etsin tutkielmaani lähteitä eri tietokannoista kuten Google Scholarista, Scopuksesta ja ACM Digital Librarysta. Hakusanoina käytin muun muassa children, programming, teaching, k-12 ja edellä mainittujen yhdistelmiä. Näiden avulla löysin hyviä lähteitä, joita lukemalla kuitenkin huomasin, että jotkin tietyt asiat jäivät uupumaan kyseisitä lähteistä. Esimerkiksi jotkin tutkimukset selostivat tarkasti tutkimuksen eri vaiheet, mutta niistä saattoi puuttua esimerkiksi tutkimustulokset tai selostus käytetystä ohjelmointiympäristöstä tai –kielestä. Aloin lopuksi hakea suoraan tiettyjen ohjelmointikielten nimillä, tarkoituksena löytää vastauksia puutteisiin tai uusia näkökulmia. Tässä vaiheessa olin valinnut ohjelmointikielet ja -ympäristöt, joita tulisin käsittelemään tutkielmassani.

Lopputyö pitää sisällään katsauksen tiettyihin ohjelmointikieliin ja –ympäristöihin, jotka soveltuvat ohjelmoinnin opettamiseen ja on suunnattu lapsille. Ohjelmointikielet ja –ympäristöt on esitelty pääpiirteittäin ja niihin liittyvien tutkimusten tuloksia käydään läpi. Esittelyn lisäksi lopputyössä on käsitelty syitä miksi ja miten ohjelmointia tulisi opettaa lapsille. Lopuksi käydään läpi johtopäätökset ja omaa pohdintaa aiheesta.

2. Lapsen kehityksen vaikutus ohjelmoinnin opetukseen

Iälle, jossa ohjelmoinnin opettaminen kannattaa aloittaa ei ole tarkkaa määritystä. Ohjelmoinnin opettamiseen vaikuttaa iän lisäksi moni muukin tekijä, kuten opettamiseen käytettävät työkalut, konteksti, opettaja, kulttuuri ja mahdollisuudet kehittää omia kykyjä. (Duncan 2014.)

2.1. Pienet lapset

Suurin osa robotiikan ja ohjelmoinnin tutkimuksista on keskittynyt myöhemmän vaiheen koulunkäyntiin (Bers 2014). On kuitenkin todettu, että jo neljävuotiaat lapset voivat oppia tietokoneohjelmoinnin peruskäsitteitä (Flannery 2103). Lisäksi jopa 4-6-vuotiaat lapset voivat oppia luomaan ja ohjelmoimaan yksinkertaisia robottiteknisiä projekteja. Robotiikka auttaa lasta kehittämään hienomotorisia taitoja, silmän ja käden koordinaatiota sekä luomaan algoritmisen ajattelun (computational thinking) taitoja. (Bers 2014.)

Pienten lasten on vaikea ymmärtää abstrakteja sisään- ja ulostulolaitteita eli näppäimistöä ja näyttöä. He tarvitsevat konkreettisia laitteita ja siksi riippuvuus näppäimistöön ja näyttöön estävät monien ympäristöjen käytön. (Wyeth 2008.)

Lapset alkavat oppia 4-6 -vuotiaana kuinka ymmärtää, esittää ja muistaa objekteja mielessään ilman, että heidän edessään olisi kyseiset objektit. Tässä ikähaarukassa heillä on vielä vaikeuksia abstraktien käsitteiden, kuten näkökulman ottaminen ja mentaalisen mallintamisen kanssa, joten heidän tarvitsee usein turvautua fyysisiin esittelyihin ja objekteihin auttaakseen heitä muotoilemaan, testaamaan ja korjaamaan heidän ajatuksiaan siitä, kuinka maailma toimii. Nuorille lapsille erityinen oppimisen väline on leikkiminen, joka auttaa tässä kehityksessä. Lisäksi lapset käyttävät heidän viittä aistiaan leikkiessään. (Strawhacker 2014.)

2.2. Vanhemmat lapset

Luonnollisen kielen oppimisen kannalta on parasta, että sen opettelu tapahtuu silloin kun on nuori. On myös todistettu, että uuden kielen opettelu puberteetin aikaan on normaalia hankalampaa. (Duncan 2014.)

Yläaste on kriittinen aika identiteetin rakentumiselle kuten myös kognitiivisen kehityksen analyyttiselle päättelylle (Grover 2015). Ylemmän peruskoulun oppilaat eroavat merkittävästi alemman peruskoulun oppilaista. Toisella ja kolmannella luokalla lukemaan oppiminen on vielä sen verran vaativaa, että lapselle jää vain vähän resursseja jäljelle käsitelläkseen sisältöä. (Hill 2015.)

Neljännestä luokasta kahdeksanteen, lapsi alkaa yhä enemmän saada uutta informaatiota tekstistä. Ylempi peruskoulu onkin tärkeä siirtymävaihe lapsen kehitykselle. Tällöin oppilaat kehittävät kielellisiä ja kognitiivisia taitojaan, joita tarvitaan onnistuneeseen vuorovaikutukseen tietokoneen kanssa. Myöhemmin lapset oppivat tärkeitä käsitteitä

muista aloista, kuten matematiikasta, joita tarvitaan ohjelmointiin. Esimerkiksi tärkeitä matematiikan käsitteitä kuten negatiivisia numeroita, jako- ja prosenttilaskuja opetetaan ylemmän peruskoulun aikana. Näin ollen tätä nuoremmilta lapsilta ei voida edellyttää näiden taitojen osaamista. (Hill 2015.)

2.3. Kehityspsykologian teoriat

Kehityspsykologia auttaa selittämään minkälaisien konseptien kanssa oppilaat pystyvät toimimaan missäkin iässä. Piaget'n kehityksen neljä vaihetta luonnehtii lapsen kehitystä siitä, miten heidän tietoisuus maailmasta muuttuu, ja kuinka heidän abstrakti ja looginen ajattelu kehittyy. Piaget'n teoriaa ei kuitenkaan enää pidetä universaalina, mutta se selittää, miten lapsen kehitysvaiheet yleensä etenevät. Piaget'n mallin mukaan 7-11-vuotiaiden psyykkiset kyvyt ovat kehittyneet siten, että he alkavat ymmärtää paremmin kuin ennen maailmasta heidän ympärillään, joka edesauttaa heitä ajattelemaan loogisesti. Lapsen kehitys 11-16-vuotiaana on riittävä varsinaisen tietokoneohjelmoinnin perusteille, jossa käytetään symboleja ja loogista päättelyä. (Duncan 2014.)

Neo-Piaget'n teoria ei pidä kognitiivista kehitystä yhtä yksinkertaisena kuin Piaget'n teoria, vaan kehitys voi riippua yksilöstä ja oppimisen/opittavasta alueesta. Teorian mukaan ohjelmoinnin konsepteja voi opettaa jopa 5-7-vuotiaille lapsille, kunhan kieli on ikäryhmälle sopivaa. (Duncan 2014.)

2.4. Sukupuolen vaikutus

Yleisesti tietojenkäsittelyä ja ohjelmointia pidetään poikavoittoisena. Tutkimuksissa on selvitetty, että tyttöoppilaat kiinnostuvat tietojenkäsittelystä ja ohjelmoinnista parhaiten ollessaan noin 12-vuotiaita. Tällöin olisi siis paras aika innostaa tyttöjäkin ohjelmointiin. Lukion ja puberteetin aikana suurin osa tyttöoppilaista tuntee heidän itsetuntonsa laskeneen tietokoneiden käyttöön, kuten ohjelmointiin, liittyen. Tässä iässä tytöt saattavat kokea leimautuvansa tietokoneen käyttäjänä tietynlaiseen muottiin. Aiempi kokemus voi kuitenkin auttaa heitä ylläpitämään heidän luottamustaan tietojenkäsittelyyn. Positiivisesta kokemuksesta tietojenkäsittelyn opinnoissa koulussa on vaikutusta tyttöoppilaisiin siten, että he jatkavat silloin tietojenkäsittelyn opintoja todennäköisemmin tulevaisuudessa. Toisaalta negatiiviset kokemukset, kuten heikko opetus, vaikuttaa myös vahvasti kiinnostukseen aiheeseen ja alaan. (Duncan 2014.)

Kulttuuri voi vaikuttaa ohjelmoinnin sukupuolineutraaliuteen. Se voi joko ehkäistä tai edistää sukupuolien välistä epätasapainoa vaikuttamalla asenteisiin ohjelmoinnin oppimisesta ja mahdollisesta työurasta. (Duncan 2014.)

3. Ohjelmoinnin opettamisen muodot

Jopa neljävuotiaat lapset voivat oppia algoritmisen ajattelun konsepteja, mikä tukee heidän kirjallisuuden, matemaattisten ja sosiaali-emotionaalisten taitojen kehitystä. Jotta olisi mahdollista opettaa algoritmista ajattelua nuorille lapsille, tarvitaan sopivia apuvälineitä, pedagogiikkaa, ja arviointeja algoritmisen ajattelun oppimisesta varhaisen lapsuuden aikana. (Portelance 2015.)

Ohjelmoinnin ja algoritmisten ratkaisujen rakentamisen oppiminen on havaittu vaikeaksi, mutta myöskään sen opettaminen ei ole kovin helppoa (Garner 2015). Ohjelmoinnin opettamiseen ei riitä, että tietää kuinka ohjelmoida, vaan siihen tarvitaan edistyksellisempiä taitoja. Useat tietokonekäsitteet täytyy pystyä esittämään ja muotoilemaan siten, että ne voidaan ymmärtää mahdollisimman helposti. (Garneli 2015.)

Kun tietokonetunnit toteutetaan erilaisilla tavoilla ja niiden lähtökohdat ovat mahdollisimman tavanomaiset, kuten tyypillisessä kouluympäristössä, oppiminen voi tehostua. Lisäksi tällainen tapa edistää oppilasta kehittämään hänen perustaitojaan, kuten algoritmista ajattelua (computational thinking) ja luovuutta. (Garneli 2015.)

Erilaisten opettamistekniikoiden testaaminen voi auttaa motivoimaan oppilaita, varsinkin niitä, jotka saavat heikkoja tuloksia koulussa. On tärkeää käyttää erilaisia tekniikoita ja harjoitteita tavanomaisen opetuksen tukena, jotta oppilas voi oppia laajemman kirjon eri oppimistyyleistä. Oppimistyylit tukevat lapsen oppimista myös tulevaisuudessa. (Garneli 2013.)

Opettamisen lähestymistapa voi vaikuttaa merkittävästi johdatuksessa ohjelmointiin ehkäisemällä ongelmia ja väärinymmärryksiä, minkä lisäksi se voi myös innostaa oppilaita. Ei ole yhtä tapaa opettaa ohjelmoinnin taitoja. Oppilaat kehittyvät ja oppivat eri tavalla, joten tarvittaessa on oltava erilaisia opettamisen muotoja vastaamaan lasten tarpeita. (Garneli 2015.)

Tärkeitä taitoja on kyky rakentaa ja luoda tarkoituksenmukaisia asioita tietokoneella. Päästäkseen tähän tavoitteeseen, olisi hyvä toteuttaa ohjelmoinnin tunteja nuorille ikäluokille peruskoulussa. Teknisen osaamisen lisäksi oppitunnit kehittäisivät algoritmista ajattelua, kriittistä ajattelua ja luovia taitoja. (Garneli 2015.)

Ikä, sukupuoli, aktiviteetin tyyppi ja tavoite on syytä huomioida, kun valitsee sopivat välineet ohjelmoinnin opettamiseen (Garneli 2015).

3.1. Ohjelmointiympäristöt

Miettiessään jotakin ongelmaa ihminen muodostaa tästä mielessään jonkinlaisen käsityksen. Kun hän syöttää tämän käsityksensä tietokoneelle, sen on oltava tietynlainen, jotta ohjelmoitu tietokone hyväksyy tämän. Ohjelmointiympäristöjen perusidea on pienentää tätä aukkoa ihmisen ja tietokoneen välillä. (Smith 1996.)

Tähän on olemassa kaksi tapaa: opettaa ajattelemaan kuten tietokoneet tai opettaa tietokoneita käsittelemään käyttäjän tuottamaa esitystä. Suorin tapa, jolla voitaisiin opettaa ohjelmointia, olisi opettaa lapsille, miten luodaan mentaalinen malli tietokoneesta. Tämä ei ole kuitenkaan erityisesti lapsille mikään ihanteellisin tapa. Ongelmana on, että vaikka lapset oppisivat käyttämään tekniikoita, he eivät silti pitäisi lopputuloksesta. Lapset eivät halua ajatella kuten tietokoneet, vaan he haluavat kontrolloida niitä. (Smith 1996.)

Tähän päivään mennessä on kehitetty useita ohjelmointiympäristöjä, jotka on suunnattu erityisesti nuorille lapsille, esimerkkeinä erilaiset toteutukset Logo-kielestä, ToonTalk ja Scratch. Lapsille suunnattujen ympäristöjen tarkoituksena on tehdä ohjelmoinnista entistä sopivampaa lapsille huomioiden heidän kehityksensä. Tällaisia seikkoja ovat esimerkiksi syntaksin yksinkertaistaminen, komentojen välitön suorittaminen ja ohjelmoinnin ajatusmallien sovittaminen metaforiksi. Lisäksi ohjelmointi voidaan yksinkertaistaa muotoon, jossa käytetään laattoja, jotka sisältävät komentosymboleita, joita lapset voivat siirtää (vetää ja pudottaa) muodostaakseen ohjelmia. (Fessakis 2013.) Seuraavissa kappaleissa tulen käsittelemään kolmea eri ohjelmointiympäristöä ToonTalkia, Stagecast Creatoria ja Logoa. Edellä mainittujen ohjelmointiympäristöjen tavoin Scratch on sekä ohjelmointikieli että –ympäristö, mutta sitä käsitellään kuitenkin vasta myöhemmässä vaiheessa.

3.1.1. ToonTalk

ToonTalk on animoitu interaktiivinen maailma, jossa on objekteja, joita lapsi voi valita itselleen ja käyttää pelissä. Tämä tarjoaa lapselle metaforia ohjelmoinnin rakentumisesta. ToonTalk on ohjelmointikieli, joka on luotu erityisesti lapsille ja jonka lähdekoodi on animoitu. ToonTalkin ohjelmointiympäristö tarjoaa visuaalisia vihjeitä auttaakseen ohjelmoijaa havaitsemaan ongelman rajauksen. Vihjeet ovat vain yksi tapa, jolla järjestelmä pyrkii auttamaan käyttäjää ymmärtämään ohjelman toteutuksen. (Morgado 2008.) Tutkimus osoitti, että lapset nauttivat arvoituksista ja oppivat joitain niiden parissa ohjelmoinnin taitoja. Arvoitusten huolellisesti suunniteltu järjestys voi olla pedagogisesti todella tehokasta. Lapsiin vetoaa paremmin arvoitukset, kun ne on sisällytetty johonkin kertomukseen, kuten esimerkiksi seikkailuun. ToonTalkissa arvoitusten ratkaiseminen sisältää ajan mittaamista, matematiikkaa ja hieman uusia ohjelmoinnin tekniikoita. ToonTalkin ikähaitari on suhteellisen laaja. Jopa kuusivuotiaat pystyvät ymmärtämään ToonTalkin toimintoja, mutta jotkin seikat saattavat tuntua hankalilta jopa 9-10-vuotiaille. ToonTalk sisältää tekstistä puheeksi –toiminnon, jonka myötä lukutaidottomat lapset voivat käyttää ohjelmaa, eivätkä tarvitse vanhempien apua. (Kahn 1999.)

On monia tapoja opettaa lapsille ohjelmointia ToonTalkin avulla. Tyypillisesti tehokkain tapa on järjestää monia kahdenkeskeisiä tapaamisia lapsen kanssa. Opettajan tulisi olla hyvin taitava ja tietävä. Tällaisia opettajia ei kuitenkaan ole tarpeeksi. Toisaalta jotkut lapset taas oppivat hyvin omatoimisella tutkiskelulla esimerkiksi ToonTalkin vapaamuotoisessa moodissa. Näitä lapsia on kuitenkin vähemmän. Tutkimuksessa todettiin, että suurin osa lapsista kysyi, mitä heidän tulisi tehdä seuraavaksi mieluummin kuin tutkisivat asiaa itse. Lapsille pulmapeli on ihanteellinen. He pitävät sitä hauskana ja haastavana, mutta heille ei tule tunnetta siitä, että olisivat hukassa. Hyvässä pelissä pelaajalle tulee illuusio siitä, että hän kontrolloi tapahtumia ja niiden kulkua, mutta todellisuudessa peli etenee, kuten sen on suunniteltu etenevän.

ToonTalk on pulmapeli jota lapset pelaavat myös huvikseen, ja jossain tapauksissa he eivät edes tiedä oppivansa ohjelmoinnin taitoja. Lisäksi koska oppilaat onnistuivat ratkaisemaan arvoituksia, voidaan osoittaa, että arvoitukset ovat onnistuneita ja lapset oppivat tietokoneohjelmointia. (Kahn 1999.)

3.1.2. Stagecast Creator

Stagecast Creator on ohjelmointiympäristö aloitteleville ohjelmoijille. Ohjelman avulla voidaan rakentaa simulaatioita. Alkujaan Stagecast Creatoria kutsuttiin nimellä KidSim, jonka jälkeen se muutettiin Cocoaksi, ja lopulta sai nykyisen nimensä. Stagecast Creatorin tavoitteena on tehdä tietokoneista hyödyllisempiä opiskelun näkökulmasta. Simulaatiot ovat tehokkaita opettamisen työkaluja, joiden ideana on tehdä abstrakteista asioista konkreettisempia ja ymmärrettävämpiä. Simulaatiot rohkaisevat kokeilemaan, auttaen lasta kehittämään järjestelmällistä päättelykykyä. (Smith 2000.)

Cocoa on ympäristö, joka mahdollistaa käyttäjän ohjelmoimalla rakentamaan ja muokkaamaan symbolista simulaatiota. Simulaatio on tietokoneella ohjailtava mikromaailma, jossa objektit liikkuvat ympäriinsä näytöllä ja ovat vuorovaikutuksissa toisiinsa. (Smith 1996.)

Cocoaa on testattu yli 300 lapsen kanssa jotka ovat olleet 5-15-vuotiaita. Jokainen heistä oppi ohjelmoimaan Cocoaa. He olivat yleensä saaneet vain noin 5-30 minuuttia ohjausta Cocoan käyttöön. Lisäksi lapset vaikuttivat pitävän ohjelmointiympäristöstä. Tutkimus osoittaa, että lapset voivat oppia ja nauttia tämän tyyppisestä ohjelmoinnista. (Smith 1996.)

3.1.3. Logo

Logo-ohjelmointiympäristöt käyttävät periaatetta, jossa käyttäjä osoittaa heidän komentonsa yksinkertaiseen ohjelmistoagenttiin, jolla on rajattu sanavarasto. Komennot on esitetty yksinkertaisina painikkeina. Ohjelmat muodostetaan painikkeiden sarjoista, jotka näkyvät näytön alareunassa. Käyttäjä voi muokata ohjelmaa lisäämällä tai poistamalla painikkeita. Hänellä on myös mahdollisuus suorittaa koko ohjelma tai jokin tietty komento. (Fessakis 2013.)

Logo-ohjelmointikielen käyttäminen kehitti 4-5-vuotiaiden lasten työskentelytapoja. Lapsista tuli entistä itseohjautuvampia ja he alkoivat hiljalleen tukeutua enemmän ikätovereihinsa ja vähemmän opettajiin. Logo johti yhteistyöhön perustuvaan käyttäytymiseen lasten välillä, ja kehitti heidän sosiaalisia ja kielellisiä taitojaan, minkä lisäksi kannusti nuoria lapsia keskittymään heidän työskentelyynsä. Tutkimus osoittaa myös, että Logolla on pedagogista vaikutusta matematiikan oppimiseen, ajattelun taitojen ja ongelmanratkaisutaitojen kehittämiseen. (Fessakis 2013.)

Tutkimuksessa lapset osallistuivat aktiivisesti osoittaen innokkuutta ja mielihyvää esitettyjen ongelmien ratkomisesta. Tutkimus osoittaakin, että käytetyt oppimistehtävät ja ohjelmointiympäristöt ovat lasten mielestä kiinnostavia ja mukaansatempaavia. Lapset saivat mahdollisuuden kehittää heidän matemaattisia taitojaan, ja sen lisäksi heidät johdateltiin tietokoneohjelmoinnin peruskäsitteisiin. (Fessakis 2013.)

3.1.4. Alice

Alice ohjelmointiympäristön tarkoituksena on opettaa ohjelmoinnin perusteita heille, joilla ei ole aiempaa kokemusta ohjelmoinnista. Ympäristön avulla käyttäjä voi luoda animaatioita kertoakseen tarinan tai pelatakseen interaktiivista peliä. Käyttäjä näkee välittömästi, miten heidän animaatio-ohjelmansa toteutuu, minkä vuoksi käyttäjä pystyy helposti ymmärtämään suhteen ohjelmalausekkeiden välillä, ja objektien käyttäytymisen heidän animaatiossaan. (Sandoval-Reyes 2011.) Alice on ohjelmointiympäristö, jota käytetään vedä ja pudota -menetelmällä. Alice sisältää monia ennalta määriteltyjä objekteja ja toimintoja, joiden avulla on tarkoitus pelien luomisen avulla opettaa olio-ohjelmoinnin alkeita lapsille. (Werner 2012.)

Tutkimus yläasteikäisistä oppilaista ohjelmoimassa pelejä käyttäen Alicea osoittaa, että oppilaat pystyvät onnistuneesti työskentelemään usean abstraktin rakenteen kanssa. Toisaalta oppilaat joutuivat ponnistelemaan muuttujien alustamisen, silmukoiden, ehtolauseiden, osoittimien ja rekursion (toiston) kanssa. Lopputulos osoittaa, että tämän ikäisille oppilaille saattaa tulla raja vastaan tietyn abstraktiotason jälkeen, ainakin ilman erilaista pedagogista lähestymistapaa. (Duncan 2014.)

3.2. Ohjelmointikielet

Ohjelmoinnin opettaminen on haastavaa ainakin kahdesta eri syystä. Ensinnäkin tavanomaisen ohjelmointikielen hallinta vaatii uuden kielen opettelun. Uuden kielen oppiminen vie usein paljon aikaa ja on suurimmalle osalle hankalaa. Toiseksi ohjelmointikielet ovat keinotekoisia kieliä, joilla on luonnollisiin kieliin verrattuna erilainen tietoteoria. Lisäksi ohjelmointikieliä käytetään vieraassa maailmassa, jossa tietokoneen tietorakenteet ja algoritmit ovat läsnä. Tämän vuoksi aloittelijan on hyvin vaikea sisäistää ohjelmointikieliä. Ongelmaan on kehitetty ratkaisu, jonka idea on tehdä ohjelmoinnista enemmän ajattelun kaltaista. (Smith 1996.)

Ohjelmointi tunnetaan sen monimutkaisuudestaan ja vaikeudestaan, ja oppilailla on ollut hankaluuksia oppia hallitsemaan vaadittuja osa-alueita. Varhainen tutkimus osoittaa, että visuaalinen ohjelmointi voi olla tehokkaampaa kuin tavallinen tekstimuotoinen ohjelmointi, koska sitä käyttäessään oppilaat ovat usein motivoituneempia, eikä ohjelmointikielten syntaksi ole silloin taakkana. (Kaucic 2011.)

Graafinen ohjelmointikieli, verrattuna tavanomaiseen, alentaa kognitiivista muuria poistaen esteitä kuten tietyn sanaston muistamisen ja alustamisen säännöt tietokoneohjelmoinnista. Tavanomaisessa ohjelmoinnissa ohjelmoijan on osattava itse tuottaa ohjelmakoodia käyttäen jonkin ohjelmointikielen ennalta määriteltyjä sääntöjä ja sanastoa. Graafiset ohjelmointikielet tarjoavat näkymän, jossa komentojen kirjoittamisen sijasta, käyttäjä näpäyttää valmiita komentoja yhteen muodostaakseen ohjelmakoodia ja näkee ohjelman tulosteen välittömästi. Graafiset ohjelmointikielet tarjoavat hyötyä kaikenikäisille uusille ohjelmoijille, jonka jälkeen on helpompi siirtyä tavanomaisiin ohjelmointikieliin kuten Javaan. (Hill 2015.)

3.2.1. Scratch ja ScrachJr

Scratch on lohkopohjainen ohjelmointikieli, joka on suunnattu erityisesti 8-16-vuotiaille lapsille (Hill 2015). Lohkopohjaisten ohjelmointikielten yksinkertaistettu käyttöliittymä vähentää kognitiivista kuormaa aloittelevilta ohjelmoijilta (Hansen 2015). Scratch ohjelmointikielellä ja ympäristöllä voidaan luoda interaktiivisia tarinoita, pelejä, animaatioita ja taideprojekteja. Scratchin kanssa ohjelmat tehdään yhdistelemällä lohkoja toisiinsa. Lohkot sisältävät ennalta määriteltyjä komentoja, joita Scratchissa on arviolta noin 90 kappaletta. (Kaucic 2011.)

Syntaksin luomat ongelmat ja kehitysympäristön monimutkaisuus on poistettu, minkä vuoksi lapsi voi keskittyä ongelmien ratkaisemiseen ja ohjausrakenteiden ja yksinkertaisten tietorakenteiden oppimiseen (Kaucic 2011).

Tutkimuksessa yläasteen 11-14-vuotiaille opiskelijoille toteutettiin seitsemän viikkoa kestävä oppijakso, jonka tarkoituksena oli motivoida ja valmistaa oppilaita algoritmiseen ongelmanratkaisuun tulevaisuuden kannalta. Opintojakso toteutettiin osittain koulun luokassa ja osittain itsenäisesti kotona. Seitsemän viikon oppijakson jälkeen lapset pystyivät käyttämään apunaan monia Scratchista (graafinen ohjelmointikieli) opittuja asioita tekstipohjaisessa ohjelmointikielessä. Lapset pystyivät laajasti tulkitsemaan ohjelmia uudessa tekstipohjaisessa ohjelmointikielessä, poikkeuksena oli joidenkin mekanismien rakenteet kuten silmukat, joita oli vaikea ymmärtää. Oppijakso vaikutti myös lasten asenteisiin, sillä se auttoi oppilaita näkemään tietojenkäsittelyn erilaisessa valossa, ja olivat innokkaita oppimaan lisää. Lasten kapea ja naiivi näkökulma muuttui huomattavasti. He näkivät oppijakson jälkeen tietojenkäsittelyn olevan tutkimusalana ongelmanratkaisua, jossa tietokoneita käytetään lähinnä välineenä tehdäkseen ihmisten elämästä helpompaa, huomioiden sekä hyödyllisen että viihteellisen puolen. (Grover 2015.)

ScratchJr on graafinen ohjelmointikieli, joka on tehty Scratchin pohjalta vastaamaan nuorempien lasten tarpeita. Suunnittelussa on huomioitu 5-7-vuotiaiden kehitys ja oppiminen (Flannery 2013). Esimerkiksi ScratchJrissa vaaditaan vähemmän matemaattista osaamista kuin Scratchissa. ScratchJrin ohjelmointikieli käyttää symboleita sanojen sijasta ja on sisällöltään suppeampi kuin Scratch. Lisäksi ScratchJrissa ei ole muuttujia eikä listoja. (Hill 2015.) Suurin osa ohjelmoinnin apuvälineistä on suunnattu lapsille, jotka ovat vähintään 7-8-vuotiaita, mutta yleensä vielä sitäkin vanhempia. ScratchJr tarjoaa sovelluksen, jonka avulla 5-7-vuotiaat voivat oppia ohjelmoinnin käsitteitä ja ongelmanratkaisua luodessaan interaktiivisia animaatioita ja kertomuksia. Testit osoittavat, että varhaisen ScratchJr –prototyypin käyttö kehitti monia lapsia huomattavasti enemmän tarkoituksenmukaisessa näpertelyssä, ohjelman oppimisessa ja pienten projektien luomisessa kuin Scratchin. ScratchJrin avulla nuoret lapset voivat oppia ongelmanratkaisutaitoja ja ohjelmoinnin perusteita, samalla lapset saavat kokemuksia tarinankerronnasta, luovasta ajattelusta, itseilmaisusta. (Flannery 2013.)

Tutkimukset ovat osoittaneet, että oppilaat jotka käyttävät Scratchia jatkavat todennäköisemmin ohjelmoinnin opiskelua kuin he jotka ovat oppineet ohjelmointikieltä tavanomaisella tavalla (Duncan 2014). Tämä saattaa johtua siitä, että lohkopohjaiset ohjelmointikielet, kuten tässä tapauksessa Scratch, voivat parantaa lapsien asenteita ja lisätä itsevarmuutta ohjelmoinnissa (Hill 2015).

3.2.2. Blockly

Blockly on lohkopohjainen ohjelmointikieli, jossa jokainen projekti on luotu JavaScriptillä, rajaten käyttäjää tiettyihin kehittäjien suunnittelemiin projekteihin. Käyttäjä ei voi siis ryhtyä ohjelmoimaan omin päin. Blocklyn projektit ovat saman tyylisiä kuin Scratchin ja ScratchJrin, käyttäjä luo ohjelmakoodia lohkoista kontrolloidakseen hahmoja. Koska kehittäjät luovat jokaisen projektin erikseen, heillä on mahdollisuus luoda uniikkeja lohkoja ja käyttöliittymäelementtejä jokaiseen projektiin. Tämän vuoksi Blocklyn projekteja voidaan tehdä kaiken ikäisille. (Hill 2015.)

3.2.3. LaPlaya

Nuoret lapset hämmentyvät helposti monimutkaisesta käyttöliittymästä. Käyttöliittymän suunnittelussa voisi huomioida pitkälle kehittyneet työkalut, jotka voitaisiin piilottaa käyttäjiltä, jotta 5-14-vuotiaat lapset eivät kompastuisi niihin ja menettäisi heidän toimintakykyään. Tämä seikka on huomioitu kehittäessä LaPlayaa, lohkopohjainen ohjelmointikieli, joka on suunniteltu täyttämään Scratchin, ScracthJrin ja Blocklyn jättämät aukot. Se on suunnattu ala-asteen 4-6-luokkalaisille, tarkoituksena opettaa algoritmista ajattelua ja tietokoneohjelmoinnin taitoja. LaPlaya vähentää koneella kirjoittamisen vaatimuksia, poistaa mahdollisuuden syntaksivirheille sekä tarjoaa erilaisia visuaalisia vihjeitä käyttäjälle. Skriptit eli komentosarjat luodaan vetämällä ja pudottamalla lohkoja, jotka edustavat komentoja. (Hill 2015, Hansen 2015.)

3.3. Pelien kehittäminen

Pelien kehittäminen voi olla viihdyttävä oppimiskokemus, joka edistää syvällistä oppimista tietojenkäsittelyn käsitteistä (Garneli 2015).

Tutkimuksessa, jossa tutkittiin 12-13-vuotiaita lapsia, havaittiin, että tietyn tyyppiset oppilaat hyötyivät vaihtoehtoisista pedagogisista tavoista. Tutkimus osoittaa, että hyötypelit auttoivat eniten niitä jotka eivät ole kiinnostuneita tavanomaisesta opetuksesta. Aktiviteettina pelien luominen voi olla tehokas ja viihdyttävä tapa oppia, mutta se vaatii vielä lisää tutkimista. (Garneli 2013.)

Leikit ja pelit, jotka sisältävät fyysisten objektien käsittelyä, jotka taas symboloivat toisia objekteja tai ideoita, auttaa lasta löytämään säännöt ja systeemit laajemmista abstrakteista konsepteista eri koulutuksellisissa alueissa (Strawhacker 2014).

Opettajat voivat hyödyntää oppilaiden kiinnostusta peleihin tehokkaamman oppimisen puolesta (Garneli 2014). Pelien kehittäminen voi onnistuneesti tukea tietojenkäsittelyn käsitteiden oppimista viihdyttävällä tavalla. Projekteihin perustuva lähestymistapa opettamiseen voi innoittaa oppilaita kehittämään syvällisemmin taitojaan. (Garneli 2015.)

3.4. Käsin kosketeltavat teknologiat

Konkreettiset (käsin kosketeltavat) käyttöliittymät tarjoavat ympäristön, joka mahdollistaa yhteistyön lasten kesken, minkä vuoksi ohjelmoinnin oppiminen voi olla

entistä houkuttelevampaa nuorten lasten näkökulmasta (Garneli 2015). Tavanomaisessa graafisessa käyttöliittymässä syöte annetaan käyttämällä näppäimistöä ja hiirtä. Käsin kosketeltavat aineelliset (tangible) käyttöliittymät mahdollistavat syöttämään digitaalista informaatiota muokkaamalla fyysistä objektia. Konkreettinen käyttöliittymä sopii erityisesti tarkoitukseen, jossa johdatellaan lapsia ohjelmointiin, koska käyttöliittymä vastaa lapsen kehityksen tasoa jonka hän pystyy ymmärtämään. Abstrakteista käsitteistä, joita nuori lapsi ei pysty ymmärtämään, tehdään ymmärrettävä. Tutkijat ovat löytäneet tavan, jolla he voivat suunnitella ohjelmointikieliä siten, että lapsen on helppo harjoittaa. Tämä tapahtuu yksinkertaistamalla kieliopin ja esittämällä mekanismit konkreettisemmin. (Strawhacker 2014.)

3.4.1. TangibleK

TangibleK yhdistää tietokoneohjelmoinnin ja robottitekniset välineet, mitkä sopivat lapsille neljästä ikävuodesta alkaen. Lapset (4-6 v.) olivat TangibleK oppijakson aikana hyvin kiinnostuneita, ja kykenivät oppimaan monia näkökantoja robotiikasta, ohjelmoinnista ja laskennallisesta ajattelusta. TangibleK –kurssi osoittaa, että kun lapsille annetaan ikään sopivia teknologioita, oppimateriaaleja, oppijaksoja ja pedagogiikkaa, lapsi voi oppia nuoresta iästä huolimatta tietokoneohjelmoinnin käsitteitä robotiikkaan sovellettuna, minkä jälkeen he voivat ottaa ensi askeleet kohti algoritmisen ajattelun kehittämistä. (Bers 2014.) Ohjelmoitavat alustat kuten robotit voivat ylläpitää lasten innostusta luovissa aktiviteeteissa, jolloin oppiminen on luonnollisempaa ja tehokkaampaa (Garneli 2014).

3.4.2. Elektroniset palikat

Elektroniset palikat ovat uusi ohjelmointiympäristö, joka on suunniteltu erityisesti nuorille lapsille noin 3-8-vuotiaille. Elektroniset palikat ovat fyysisiä ja pinottavia, jotka ovat toiminnallisuudeltaan joko sensori-, toiminta- tai logiikkapalikoita. Yhdistelemällä palikoita, lapset voivat ohjelmoida laajasti rakenteita, jotka ovat vuorovaikutuksissa sekä toisiinsa että ympäristöönsä. Palikat tarjoavat nuorille lapsille mahdollisuuden ohjelmointiin ja dynaamisen käyttäytymisen tarkkailuun ilman monimutkaisia symbolisten merkintätapojen osaamista. Syntaktisia ja semanttisia ongelmia perinteisistä ohjelmointikielistä on siis tässä ohjelmointiympäristössä vähennetty. (Wyeth 2008.)

Elektroniset palikat toimivat hyvänä oppimisympäristönä pienille lapsille, koska ne eivät monen muun ohjelmointiympäristön tavoin ole riippuvaisia näppäimistöstä eikä näytöstä. Nämä abstraktit sisään-ja ulostulolaitteet tekevät ohjelmointiympäristöstä hankalan käyttää pienen lapsen näkökulmasta. Palikoiden konkreettisemmat sisään-ja ulostulot ovatkin tästä syystä parempia pienille lapsille käytettävyyden kannalta. (Wyeth 2008.)

3.4.3. Paperille ohjelmointi

Tietokoneet ja siihen liittyvät laitteet ja sensorit ovat tähän päivään mennessä suunniteltu ja kehitetty riittävän pieniksi, jotta niitä voi käyttää esimerkiksi paperille ohjelmointiin. Paperille ohjelmointi on enemmän arkikielistä kuin tavanomainen ohjelmointi. Paperille ohjelmointi toimii siten, että laitteita (esimerkiksi prosessori, patteri, ledi, sensori, moottori) yhdistetään toisiinsa voimakkailla materiaaleilla, kuten johtavalla tekstiilillä, magneeteilla ja johtavalla maalilla. Itse ohjelma kirjoitetaan

tietokoneella käyttäen apuna Arduino -järjestelmää, minkä jälkeen se ladataan suorittimeen. Arduino on mikro-ohjainalusta ja ohjelmointiympäristö, jonka tarkoituksena on yksinkertaistaa interaktiivisten sovellusten tai objektien luomisen. (Noble 2009.)

Laitteet (sisältää suorittimen) sommitellaan paperille tai muulle alustalle kuten paidalle, minkä jälkeen alusta koristellaan johtavalla materiaalilla siten, että saadaan toimiva ohjelma, eli laitteet ovat yhdistettynä toisiinsa. Lasten kannalta toiminta voi tapahtua siten, että heitä varten on tehty valmis ohjelma, ja heidän tehtävänään on luoda toiminnallisuus alustalle ohjelman mukaisesti. Vaihtoehtoisesti lapsille on voitu tehdä valmis alusta, johon heidän tulisi suunnitella ja kirjoittaa ohjelma Arduinolla. (Eisenberg 2009.)

3.5. Muut

Puettavaa tietotekniikkaa on lähiaikoina kehitelty. Se on lähestymistapa, joka yhdistää elektroniikan ja tietokoneet johonkin materiaaliin, kuten kankaaseen. Tämän tyyppinen teknologia, joka yhdistää ohjelmoinnin mahdollisuuden fyysisiin välineisiin, voi tehdä ohjelmoinnista entistä innoittavamman. (Garneli 2014.)

Mobiililaitteet, jotka ovat käsin kosketeltavaa teknologiaa, ovat yleistyneet ja saaneet suosiota lasten arjessa. Yksi tapa, jolla voitaisiin tehdä opetuksesta innostavampaa lasten näkökulmasta, on opettaa ohjelmointia näiden laitteiden (tablettitietokoneet, älypuhelimet) avulla. (Garneli 2014.)

3.6. Haasteita

Ohjelmoinnin opettamista nuorille lapsille tulisi suunnitella huolella, sillä eri ikäisinä lapset suhtautuvat eri lailla asioihin. Mikäli nuorelle lapselle tarjoaa liian vaikeaa tehtävää, lapsi saattaa turhautua ja kuvitella, että ei ole tarpeeksi älykäs tehtävää varten. Aktiviteettien tulisi olla tarkoituksenmukaisia ja haastavia, mutta myös saavutettavissa olevia, jotta vältytään lannistamasta lasta. (Fessakis 2013.)

Esimerkkinä opettamisen työkalujen valitsemisen haastavuudesta kertoo tutkimus, jossa on testattu, miten lohkopohjaiset ohjelmointikielet, kuten Scratch, soveltuisivat 9-12-vuotiaille. Huomioitavaa on, että Scratch on suunniteltu 8-16-vuotiaille. Scratchin rajapinnan toiminnallisuus ja matemaattinen sisältö oli kuitenkin liian kehittynyttä 9-12-vuotiaille lapsille. Samassa tutkimuksessa tutkittiin ScratchJrin soveltuvuutta saman ikäluokan lapsille. ScratchJr on kehitetty Scratchin pohjalta nuoremmille lapsille, mutta tämäkään ohjelmointikieli ei sovellu 9-12-vuotiaille. Tässä tapauksessa lapset olivat tapaukseen liian vanhoja, jolloin tehtävät eivät tarjonneet riittävästi haastetta. Ongelmallista kummassakin kielessä on rajapintojen mallit, kun työstetään esitäytettyä projektia. (Hill 2015.)

Ohjelmoinnin pedagogiikan tutkiminen on tärkeää, koska siitä ei ole vielä tarpeeksi tietoa. On tärkeää pystyä ymmärtämään teoreettinen ja käytännöllinen tapa, miten lapset ymmärtävät ohjelmoinnin, ja mikä on paras tapa opettaa sitä. (Garneli 2014.)

4. Johtopäätökset

Tutkielmassa pyrittiin selvittämään mitä ohjelmoinnin työkaluja voisi käyttää opettaessa ohjelmointia lapsille, ja syitä miksi ohjelmoinnin taitoja tulisi oppia. Lapsille on kehitetty useita erilaisia ohjelmointiympäristöjä ja –kieliä, joista tutkielmassa käsitellään Scratchia, ScratchJria, ToonTalkia, Stagecast Creatoria, Alicea, Blocklya, LaPlayaa ja Logoa. Ohjelmointiympäristöt helpottavat lasta ymmärtämään ohjelmointia konkretian avulla sekä vähentää tarvetta sääntöjen ja kielen opetteluun. Lisäksi ohjelmointiympäristöjen helposti ymmärrettävän rakenteen ja vihjeiden avulla lapset pystyvät toimimaan helpommin itsenäisesti tai ryhmissä, vähentäen opettajan tarvetta ohjata (taulukko 1.).

Nykyisen tiedon valossa voidaan olla sitä mieltä, että ohjelmoinnin opettaminen lapsille olisi hyvin suotavaa ja kannattavaa. Ei vaan ohjelmointitaitojen kehittymisen vuoksi, vaan myös lapsen ajattelun kehittymisen kannalta. Tutkimukset osoittivat, että ohjelmoinnin opettelu tuki lapsen ajattelun kehitystä kuten algoritmista ajattelua, jossa ongelmat pilkotaan pienemmiksi osiksi. Algoritminen ajattelu sisältää niitä ajattelun taitoja, joita lapset tulevat tarvitsemaan tulevaisuudessa riippumatta uravalinnasta. Ohjelmoinnin sisällyttäminen opetussuunnitelmaan on siksi järkevä ja kannattava ratkaisu. Nyt on kuitenkin mietittävä mikä olisi oikea tapa opettaa ohjelmointia. Vaihtoehtoja on monia, eikä sitä yhtä oikeaa ole olemassa. Uudet lähestymistavat, kuten pelit, voivat jopa innostaa ja motivoida lapsia oppimaan, ja niiden avulla voidaan selkeyttää lapsille hankalia aiheita esimerkiksi abstrakteja asioita (taulukko 1.).

Lapsen ikä on tärkeä huomioon otettava asia ohjelmoinnin opettamistapaa valittaessa. Pienemmillä lapsilla ajattelu ei ole vielä kehittynyt niin abstraktille tasolle, että kaikkia ohjelmointiin tarvittavia keinoja olisi käytettävissä. Ohjelmointiin tarvitaan myös muiden aineiden, kuten matematiikan, osaamista, mikä vaikeuttaa pienten lasten ohjelmoinnin oppimista. Siksi varsinainen ohjelmoinnin opettelu voidaan aloittaa vasta kun lapsi on muussa kehityksessään ja taidoissaan vaadittavalla tasolla. Kuitenkin ohjelmoinnissa tarvittavaa ajattelumallia voidaan opettaa jo jopa 4-vuotialle.

Ohjelmointiin tarvittavaa ajattelumallia voitaisiinkin alkaa opettaa jo päiväkodissa ja esikoulussa. Päiväkodeissa voisi olla esimerkiksi elektronisia palikoita, joilla lapset voisivat leikkiä ja toteuttaa ohjelmia. Kun ajattelu olisi kehittynyt jo ennen koulun alkua, olisi koulussa helpompi alkaa opetella itse ohjelmointia.

Ohjelmoinnin opettaminen koetaan haastavaksi ja siihen on kehitetty useita apuvälineitä, joita on suositeltavaa käyttää. Niiden on osoitettu auttavan lapsia oppimaan ohjelmointia ja tekevän ohjelmoinnista viihdyttävää ja mukavaa. Aiempi tietämys helpottaa ja edistää uuden tiedon luomista. Ohjelmointiympäristöt, kuten Scratch, helpottavat lasta siirtymään myöhemmin tavanomaisiin tekstipohjaisiin ohjelmointikieliin. Tutkimukset osoittavat (taulukko 1.), että lapset voivat samalla sekä viihtyä ohjelmoinnin parissa että hyötyä siitä. Ohjelmointi kehittää lapsen ajattelua, mitä tämä voi hyödyntää arjessa sekä eri oppiaineissa kuten matematiikassa.

Ohjelmoinnin opettamisessa opettajan rooli on suuri. Opettajan asenne ja tietämys ohjelmointiin vaikuttaa lapseen vahvasti. Huonot pedagogiset mallit voivat johtaa negatiivisiin asenteisiin lasten keskuudessa, minkä takia opetuksesta saattaa olla

enemmän haittaa kuin hyötyä. Ohjelmoinnin opettaminen tulee opetussuunnitelmaan syksyllä 2016 osana matematiikkaa. Suurin osa opettajista ei ymmärrä ohjelmointia. Haasteena onkin saada peruskoulujen opettajille tarpeeksi kattava tietotaito, jotta he pystyvät opettamaan ohjelmointia.

Ohjelmoinnin opettamisen keinoja on monia, mikä on hyvä asia opettamisen monimuotoisuuden kannalta. Opettamisen tulisi olla vaihtelevaa ja virikkeellistä. Tämä tukee mahdollisimman monen lapsen oppimista, koska jokainen lapsi oppii omalla tavalla. Opettamismuotojen vaihtelu myös pitää lasten mielenkiintoa yllä ja kehittää ajattelua moniulotteisemmin.

Ohjelmoinnin opettaminen edistää lapsen kehitystä. Sen ensisijaisena tavoitteena ei ole luoda kaikista lapsista ohjelmoijia, vaan taitavia ajattelijoita. Ohjelmoinnissa hankalat ongelmat puretaan pienemmiksi osiksi, mikä helpottaa kokonaisuuden hahmottamisessa.

Tutkielma rajoittuu vain tiettyihin ohjelmoinnin työkaluihin ja tutkimuksiin, minkä vuoksi se ei ole kaiken kattava. Osassa tutkimuksia oli hyvin vähän osallistujia, minkä takia tuloksia ei kannata yleistää liikaa.

Tutkielmasta voivat hyötyä esimerkiksi opettajat, lasten vanhemmat ja aiheesta tutkimusta tekevät henkilöt, sillä tutkielma käsittelee usean tutkimuksen tuloksia ja niissä käytettyjä ohjelmoinnin työkaluja. Tutkielma tarjoaa vaihtoehtoja, joiden avulla voidaan luoda pohjaa ohjelmoinnin opettamiselle lapsille. Tutkielmassa esiteltyjen ohjelmoinnin opetuksen työkaluista on tämän kappaleen jälkeen taulukko, johon on tiivistetty niiden tärkeimpiä löydöksiä. Taulukon aineisto perustuu työkaluihin, joita haluan painottaa tässä tutkielmassa kuitenkaan muita vaihtoehtoja vähättelemättä.

Jatkotutkimukset voisivat kohdistua ohjelmoinnin opettamisen hyötyihin tulevaisuuden kannalta. Koska ohjelmoinnin opetus sisältyy syksystä alkaen Suomen peruskoulujen opetussuunnitelmaan, voitaisiin tutkia kasvaako kiinnostus ohjelmointiin ja tietojenkäsittelyyn sen myötä. Lisäksi voitaisiin tutkia, miten ohjelmoinnin opettaminen vaikuttaa lasten koulumenestykseen ja oppimiseen muissa aineissa.

Taulukko 1. *ohjelmoinnin opettamisen työkaluja*

Lähde	Työkalu	Tutkimuksen ikähaarukka	Tutkittavien määrä	Tärkeimmät löydökset
Duncan (2014)	Alice	Yläaste	-	Pelien kehittämisen ohella lapset pystyivät työskentelemään onnistuneesti usean abstraktin käsitteen kanssa.
Flannery (2013)	ScratchJr	5-7-vuotiaat	-	Kehittää tarinankerronnassa, luovassa ajattelussa, itseilmaisussa, matemaattisessa päättelykyvyssä
Garneli (2013)	Scratch	12-13-vuotiaat	80 henkilöä	Pelien kehittäminen on motivoiva ja tehokas tapa oppia ohjelmointia
Grover (2015)	Scratch	11-14-vuotiaaat	-	Kehittää algoritmisessa ajattelussa ja ymmärryksestä tietojenkäsittelyyn
Hansen (2015)	LaPlaya	9-10-vuotiaat	92 henkilöä	Vähentää syntaksivirheiden mahdollisuutta ja tarjoaa visuaalisia vihjeitä. Ohjelmakoodi on näkyvillä ohjelman ajon aikanakin, minkä avulla lapsi voi verrata ajon aikaista toimintaa tuottamaansa koodiin. Toimii vedä ja pudota -menetelmällä.
Hill (2015)	ScratchJr	9-12-vuotiaat (tarkoitettu 5-7-vuotiaille)	15 eri luokkaa 5 eri koulusta	Sopii nuoremmille lapsille, Sisältö ja rajapinta liian helppo tämän tutkimuksen koehenkilöille.
Hill (2015)	Blockly	9-12-vuotiaat	15 eri luokkaa 5 eri koulusta	Uniikit projektit, voidaan määrittää tietylle iälle sopivia tehtäviä, projektien luominen vaatii osaamista
Hill (2015)	LaPlaya	9-12-vuotiaat	15 eri luokkaa 5 eri koulusta	Tarjoaa laajalti eri tyyppisiä tehtäviä ja projekteja, soveltuu kyseiselle ikähaarukalle
Hill (2015)	Scratch	9-12-vuotiaat (tarkoitettu 8-16-vuotiaille)	15 eri luokkaa 5 eri koulusta	Helpottaa siirtymistä tekstipohjaisiin ohjelmointikieliin kuten Javaan, matemaattinen sisältö osittain liian kehittynyttä
Kahn (1999)	ToonTalk	Neljäsluokkalaiset	48 henkilöä	Lukutaidottomatkin lapset voivat käyttää ToonTalkia. Lähdekoodi on animoitu. TooTalk on pulmapeli, joka tarjoaa viihdyttävän tavan oppia ohjelmointia.
Kaučič (2011)	Scratch	8-16-vuotiaat	32 henkilöä	Scratchin projektien avulla lapset oppivat tärkeitä matemaattisia ja

				algoritmisia näkökulmia.
Morgado (2008)	ToonTalk	-	-	Sisältää visuaalisia vihjeitä, jotka opastavat virhetilanteissa ja auttavat ymmärtämään ohjelman toteutusta.
Sandoval-Reyes (2011)	Alice	-	-	Alice on suunnattu niille, joilla ei ole aiempaa kokemusta ohjelmoinnista. Lapset näkevät välittömästi miten heidän ohjelmansa toteutuvat, minkä vuoksi he voivat helposti ymmärtää yhteyden ohjelmalausekkeiden välillä.
Smith (2000)	Stagecast Creator	-	-	Toimii vedä ja pudota –menetelmällä. Tekee abstrakteista asioista konkreettisempia, minkä vuoksi lasten on helpompi ymmärtää niitä ja oppia ohjelmoinnin käsitteitä.
Smith (1996)	Cocoa (Stagecast creator)	5-15-vuotiaat	Yli 300 henkilöä	Jokainen henkilö on oppinut ohjelmoimaan Cocoalla lyhyessä ajassa. Lapset ovat nauttineet tämän tyylisestä tavasta oppia ohjelmointia.
Wyeth (2008)	Elektroniset palikat	7-8-vuotiaat	12 henkilöä	Not ja and -toiminnallisuuden käyttö haastavaa, luo nuorille lapsille mahdollisuuden harjoittaa ohjelmointia ymmärrettävässä muodossa

Lähteet

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, *72*, 145-157.

Duncan, C., Bell, T., & Tanimoto, S. (2014, November). Should your 8-year-old learn coding?. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (pp. 60-69). ACM.

Eisenberg, M., Elumeze, N., MacFerrin, M., & Buechley, L. (2009, June). Children's programming, reconsidered: settings, stuff, and surfaces. In *Proceedings of the 8th International Conference on Interaction Design and Children* (pp. 1-8). ACM.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, *63*, 87-97.

Flannery, L. P., Silverman, B., Kazakoff, E. R., Bers, M. U., Bontá, P., & Resnick, M. (2013, June). Designing ScratchJr: Support for early childhood learning through computer programming. In *Proceedings of the 12th International Conference on Interaction Design and Children* (pp. 1-10). ACM.

Garneli, B., Giannakos, M. N., Chorianopoulos, K., & Jaccheri, L. (2013). Learning by Playing and Learning by Making. In *Serious Games Development and Applications* (pp. 76-85). Springer Berlin Heidelberg.

Garneli, V., Giannakos, M. N., Chorianopoulos, K., & Jaccheri, L. (2015). Serious game development as a creative learning experience: lessons learnt. In *Proceedings of the Fourth International Workshop on Games and Software Engineering* (pp. 36-42). IEEE Press.

Garneli, V., Giannakos, M. N. & Chorianopoulos, K. (2015). Computing education in K-12 schools: A review of the literature. *IEEE Global Engineering Education Conference, EDUCON, 2015-April* 543-551. doi:10.1109/EDUCON.2015.7096023

Garneli, V. (2014). Instructional Media and Teaching Methods for Engaging Children with Computer Programming. In *2014 IEEE 14th International Conference on Advanced Learning Technologies (ICALT)* (pp. 768-770). IEEE.

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, *25*(2), 199-237.

Hansen, A. K., Dwyer, H. A., Hill, C., Iveland, A., Martinez, T., Harlow, D., & Franklin, D. (2015, June). Interactive design by children: a construct map for programming. In *Proceedings of the 14th International Conference on Interaction Design and Children* (pp. 267-270). ACM.

Hill, C., Dwyer, H. A., Martinez, T., Harlow, D., & Franklin, D. (2015, February). Floors and Flexibility: Designing a programming environment for 4th-6th grade classrooms. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 546-551). ACM.

Kahn, K. (1999). A Computer Game To Teach Programming. In: Spotlight on the Future, NECC '99. National Educational Computing Conference Proceedings (20th, Atlantic City, NJ, June 22-24, 1999).

Kaučič, B., & Asič, T. (2011, May). Improving introductory programming with Scratch?. In *MIPRO, 2011 Proceedings of the 34th International Convention* (pp. 1095-1100). IEEE.

Liukas L. (2015) Hello Ruby: Adventures in coding. Otavan kirjapaino Oy, Keuruu.

Morgado, L., & Kahn, K. (2008). Towards a specification of the ToonTalk language. *Journal of Visual Languages & Computing*, *19*(5), 574-597.

Mykkänen J., & Liukas, L. (2014): Koodi2016 ensiapu ohjelmoinnin opettamiseen peruskoulussa. http://www.koodi2016.fi/. Lainattu 7.2.2016.

Noble, J. (2009). *Programming Interactivity: A Designer's Guide to Processing, Arduino, and Openframeworks*. O'Reilly Media, Inc, Sebastopol.

Portelance, D. J., & Bers, M. U. (2015, June). Code and tell: assessing young children's learning of computational thinking using peer video interviews with ScratchJr. In *Proceedings of the 14th International Conference on Interaction Design and Children* (pp. 271-274). ACM.

Sandoval-Reyes, S., Galicia-Galicia, P. & Gutierrez-Sanchez, I. (2011). Visual learning environments for computer programming. *Proceedings - 2011 IEEE Electronics, Robotics and Automotive Mechanics Conference, CERMA 2011,* 439-444. doi:10.1109/CERMA.2011.76

Smith, D. C., Cypher, A. & Tesler, L. (2000). Novice programming comes of age. Communications of the ACM, 43(3), 75-81.

Smith, D. C., Cypher, A., & Schmucker, K. (1996). Making programming easier for children. *Interactions*, *3*(5), 58-67.

Strawhacker, A. & Bers, M. U. (2014). "I want my robot to look for food": Comparing Kindergartner's programming comprehension using tangible, graphic, and hybrid user interfaces. *International Journal of Technology and Design Education,* doi:10.1007/s10798-014-9287-7

Tirronen, Mikko, (2011). IT-englannin sanakirja. OY Finn Lectura AB, Helsinki.

Werner, L., Campe, S., & Denner, J. (2012, February). Children learning computer science concepts via Alice game-programming. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 427-432). ACM.

Wyeth, P. (2008). How young children learn to program with sensor, action, and logic blocks. *The Journal of the learning sciences*, *17*(4), 517-550.